Contract-Based Integration of Cyber-Physical Analyses

Ivan Ruchkin
Dionisio De Niz
Sagar Chaki
David Garlan

October 14, 2014
14th International Conference on Embedded Software

Carnegie Mellon
isr institute for SOFTWARE RESEARCH
Software Engineering Institute

Report Documentation Page

Form Approved
OMB No. 0704-0188

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE **07 NOV 2014**	2. REPORT TYPE **N/A**	3. DATES COVERED **-**

4. TITLE AND SUBTITLE **Contract-Based Integration of Cyber-Physical Analyses**	5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) **Dionisio de Niz Ivan Ruchkin; Sagar Chaki; David Garlan**	5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR’S ACRONYM(S) 11. SPONSOR/MONITOR’S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**SAR**	**28**	

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

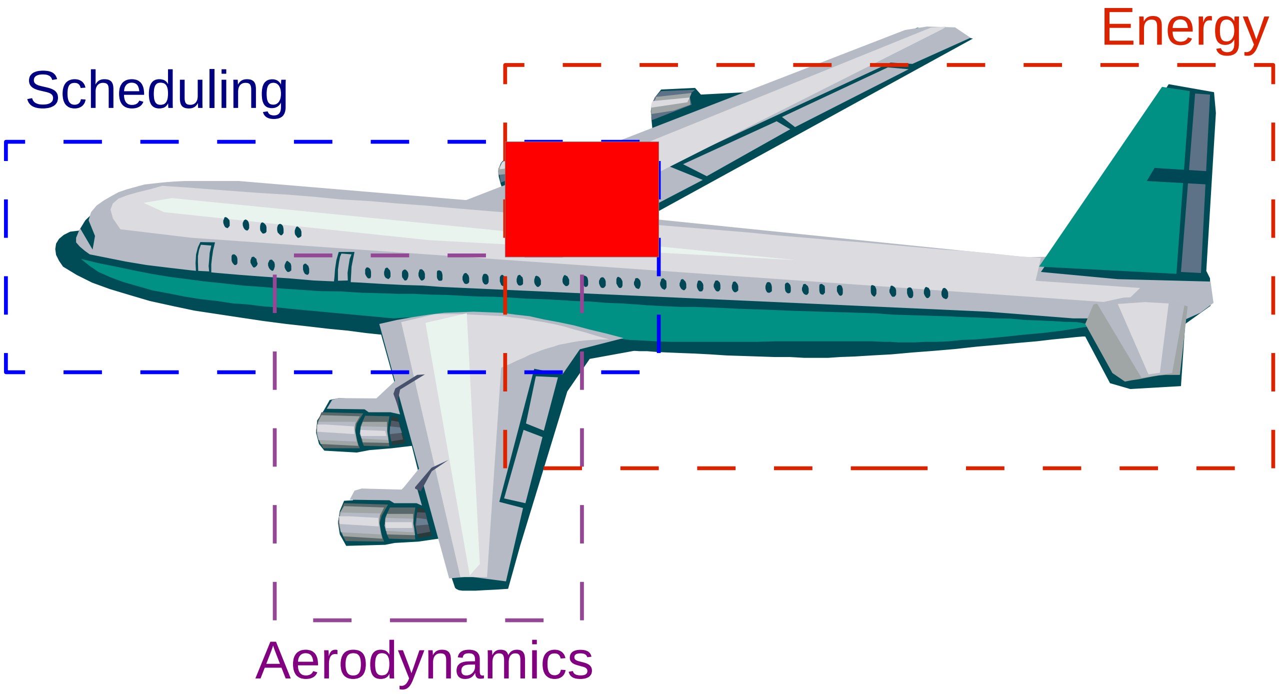
NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN “AS-IS” BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

DM-0001714

Outline

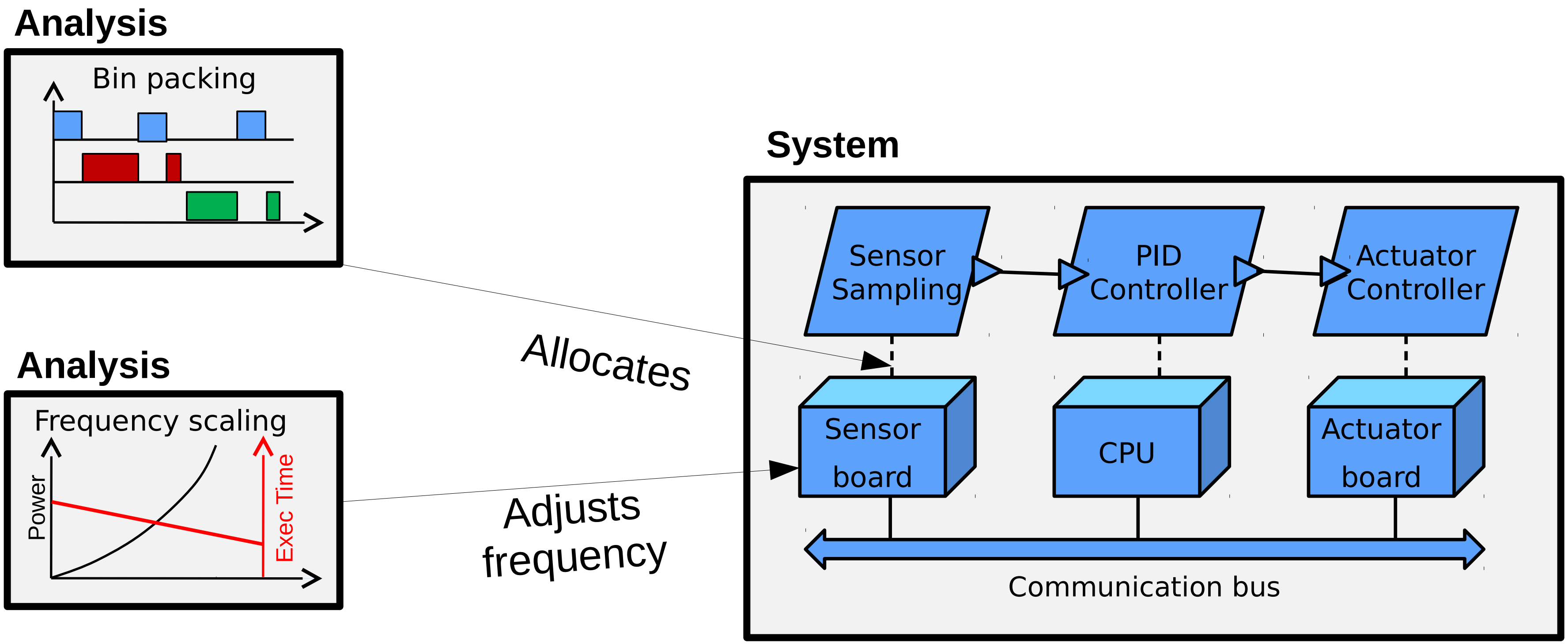
- Analysis integration problem
- Analysis contracts approach
  - Specification
  - Verification
- Experimental results

Model integration in CPS



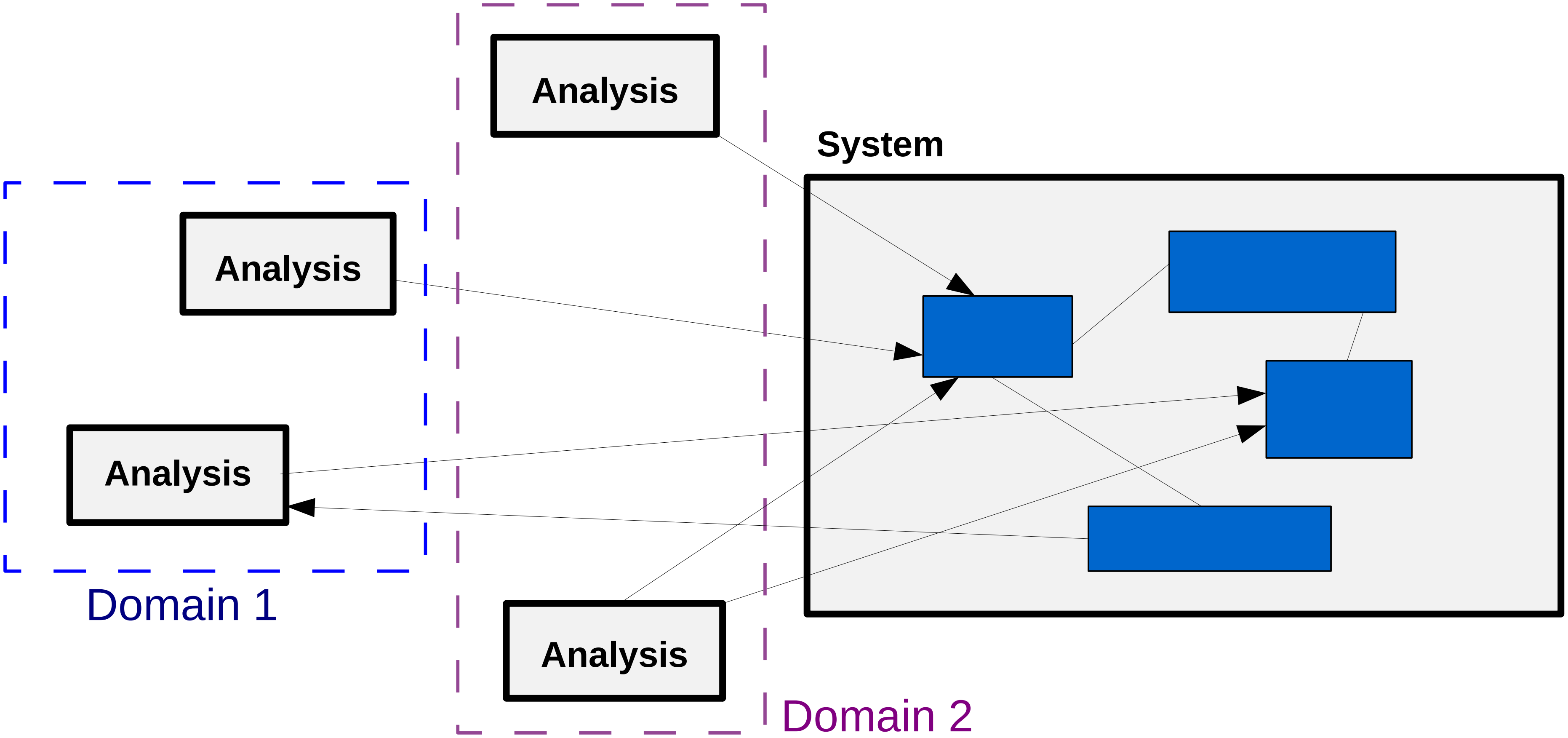
- Subtle mismatches between technical domains
- Lead to costly fixes or failures

Analytic aspect of integration



- Frequency scaling is applicable only when:
  - used after Bin packing
  - the system is behaviorally deadline-monotonic
- Otherwise, frequency scaling may render the system not schedulable
- Hence, model consistency is not sufficient

Analysis integration problem



- Out-of-order execution
- Invalidation of assumptions

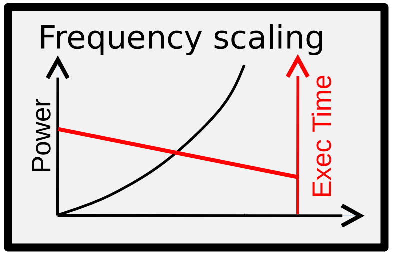
Existing solutions

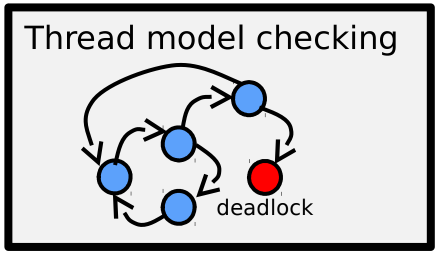
- Assume-guarantee component composition does not handle analytic integration of tools [1][2].
- Architectural views tackle model consistency, not analytic tool consistency [3][4]
- Meta-level AADL languages do not allow domain-specific semantics [5]
- Previous work on contracts: single domain only, unsound and incomplete verification [6]

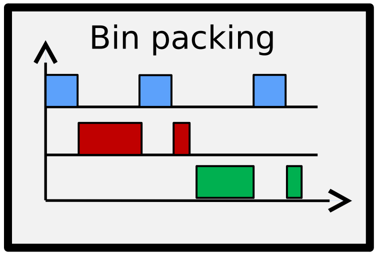
[1] Frehse et al. Assume-guarantee reasoning for hybrid I/O-automata by over-approximation of continuous interaction, 2004
[2] Sangiovanni-Vincentelli et al. Taming Dr. Frankenstein: contract-based design for cyber-physical systems, 2013
[3] Torngren et al. Integrating viewpoints in the development of mechatronic products, 2013
[4] Rajhans et al. Supporting heterogeneity in cyber-physical systems architectures, 2014
[5] Boddy et al. The FUSED meta-language and tools for complex system engineering, 2011
[6] Nam et al. Resource allocation contracts for open analytic runtime models, 2011

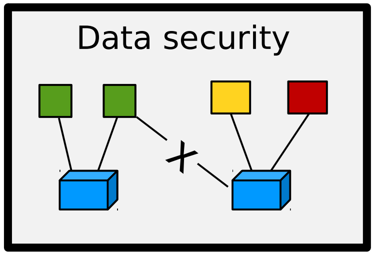
Running example

Scheduling

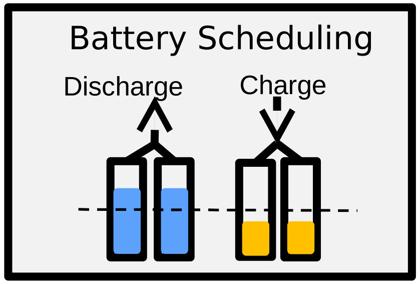


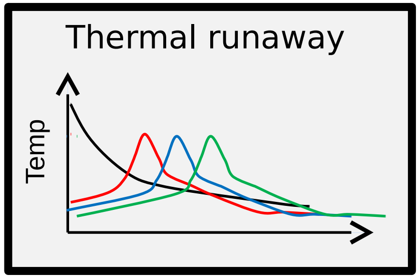




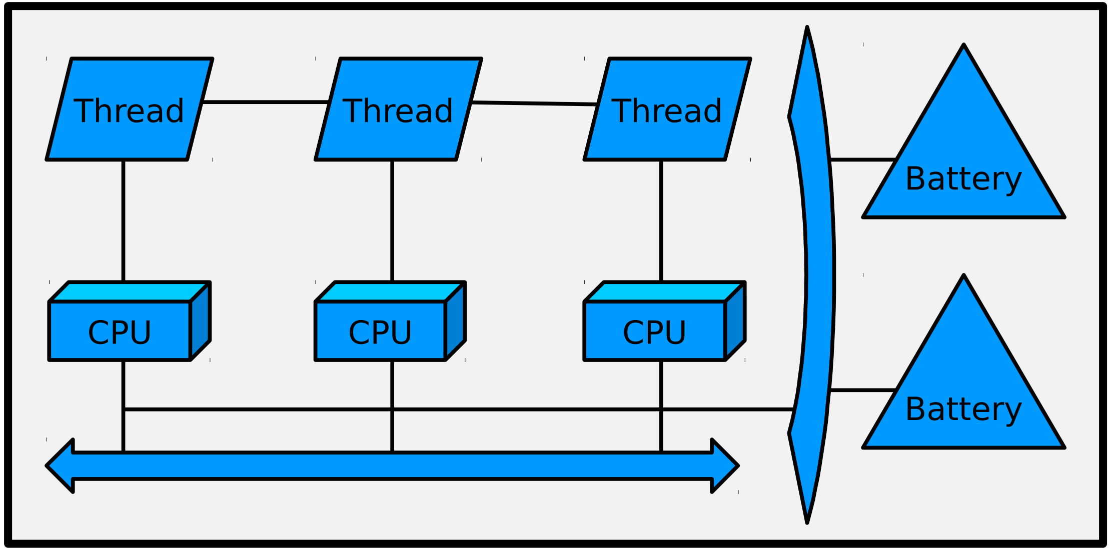


Battery





System



Outline

- Analysis integration problem
- **Analysis contracts approach**
  - Specification
  - Verification
- Experimental results

Analysis contracts approach

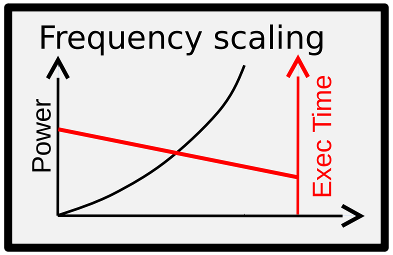
- Formalize analysis domains
- Specify dependencies and assumptions of analyses
- Determine correct ordering of analyses
- Verify assumptions of analyses

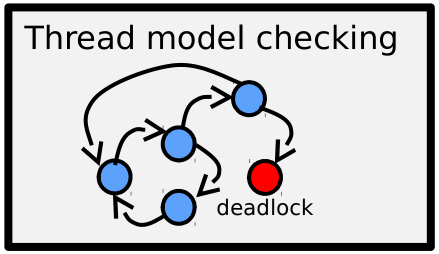
Outline

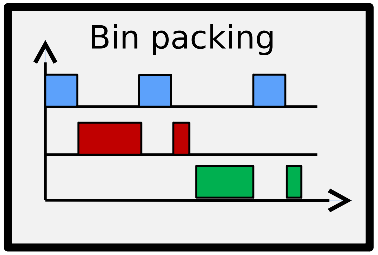
- Analysis integration problem
- Analysis contracts approach
  - **Specification**
  - Verification
- Experimental results

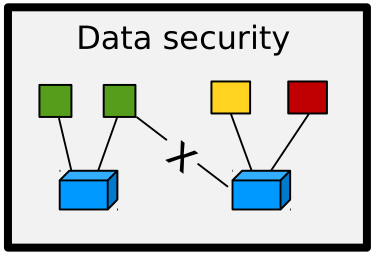
Running example

Scheduling

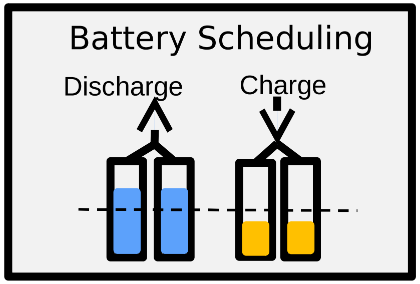

Frequency scaling

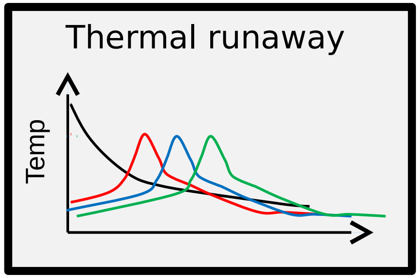

Thread model checking


Bin packing

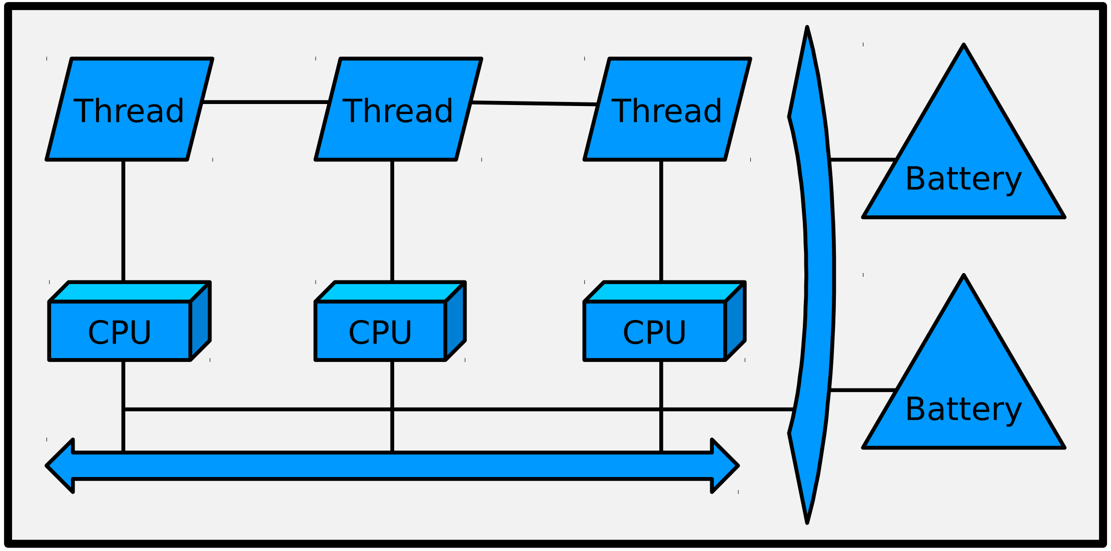

Data security

Battery


Battery Scheduling

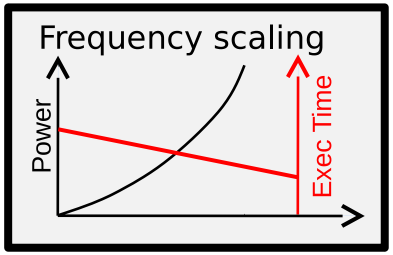

Thermal runaway

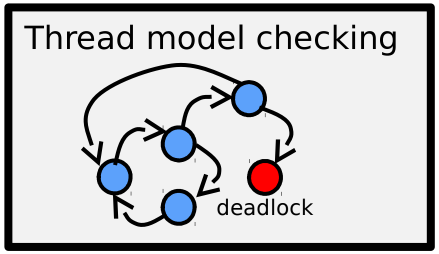
System

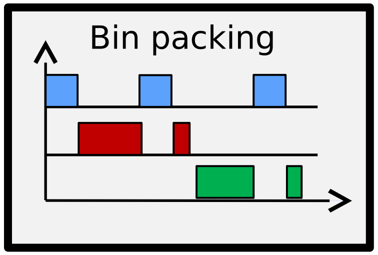


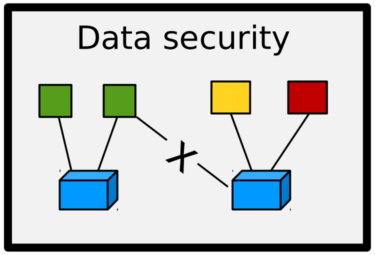
Verification domain

Scheduling domain $\boldsymbol{\sigma}_{\textbf{sched}}$

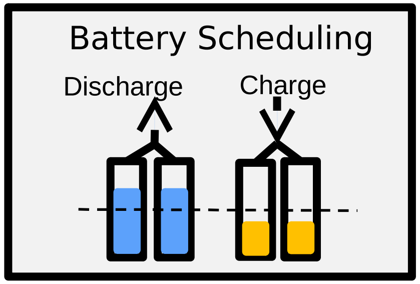


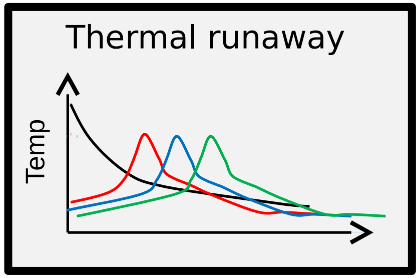




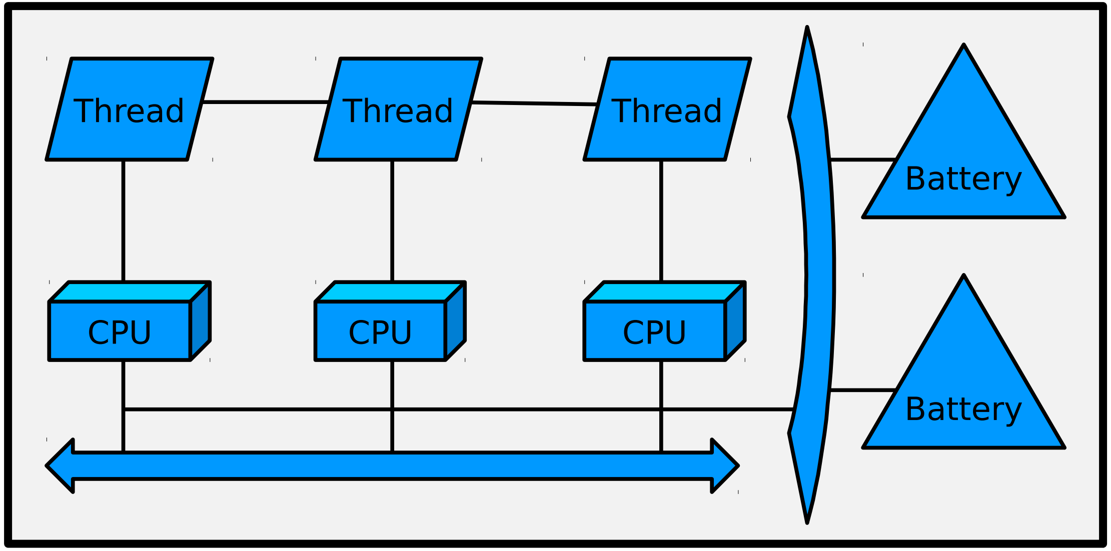


Battery domain $\boldsymbol{\sigma}_{\textbf{batt}}$





System



Verification domain

- Domain **σ** is a many-sorted signature ($\mathcal{A}$, $\mathcal{S}$, $\mathcal{R}$, $\mathcal{T}$, $\{\!\!\{\ \}\!\!\}_{\sigma}$):
  - $\mathcal{A}$: set of sorts – system elements and standard sorts
    - E.g.: $\mathbb{B}$, $\mathbb{Z}$, *Threads*, *Batteries, SchedPol*
  - $\mathcal{S}$: $\mathcal{A}_i$ x … x $\mathcal{A}_n$ → $\mathcal{A}_k$ – static functions that encode design properties
    - E.g.: *Period, Dline, CPUBind, Voltage*
  - $\mathcal{R}$: $\mathcal{A}_i$ x … x $\mathcal{A}_n$ → $\mathcal{A}_k$ – runtime functions that encode dynamic properties
    - E.g.: C*anPrmpt*: *Threads* x *Threads* → $\mathbb{B}$
      TN: *Batteries* x $\mathbb{Z}$ → $\mathbb{Z}$

Verification domain

- Domain $\boldsymbol{\sigma}$ is a many-sorted signature ($\mathcal{A}$, $\mathcal{S}$, $\mathcal{R}$, $\mathcal{T}$, $\llbracket\,\rrbracket_{\boldsymbol{\sigma}}$):
  - $\mathcal{T}$: execution semantics – set of sequences of $\mathcal{R}$ assignments
    - E.g.: thread scheduler state model for $\boldsymbol{\sigma}_{\textbf{sched}}$
      battery state model for for $\boldsymbol{\sigma}_{\textbf{batt}}$
  - $\llbracket\,\rrbracket_{\boldsymbol{\sigma}}$: domain interpretation for $\mathcal{A}$ and $\mathcal{S}$
    - E.g.: $\llbracket \mathit{SchedPol} \rrbracket_{\boldsymbol{\sigma}}$ = {RMS, DMS, EDF}
- Architectural model **m** is an interpretation $\llbracket\,\rrbracket_{\mathbf{m}}$ of $\mathcal{A}$, $\mathcal{S}$, and $\mathcal{T}$
  - E.g.: $\llbracket \mathit{Threads} \rrbracket_{\mathbf{m}}$ = { SensorSample, Ctrl_1, Ctrl_2 }
    $\llbracket \text{CPUBind} \rrbracket_{\mathbf{m}}$ **=** { (Ctrl_1, CPU_1), (Ctrl_2, CPU_2), … }
  - $\llbracket\,\rrbracket_{\boldsymbol{\sigma}} \cup \llbracket\,\rrbracket_{\mathbf{m}}$ is a full interpretation

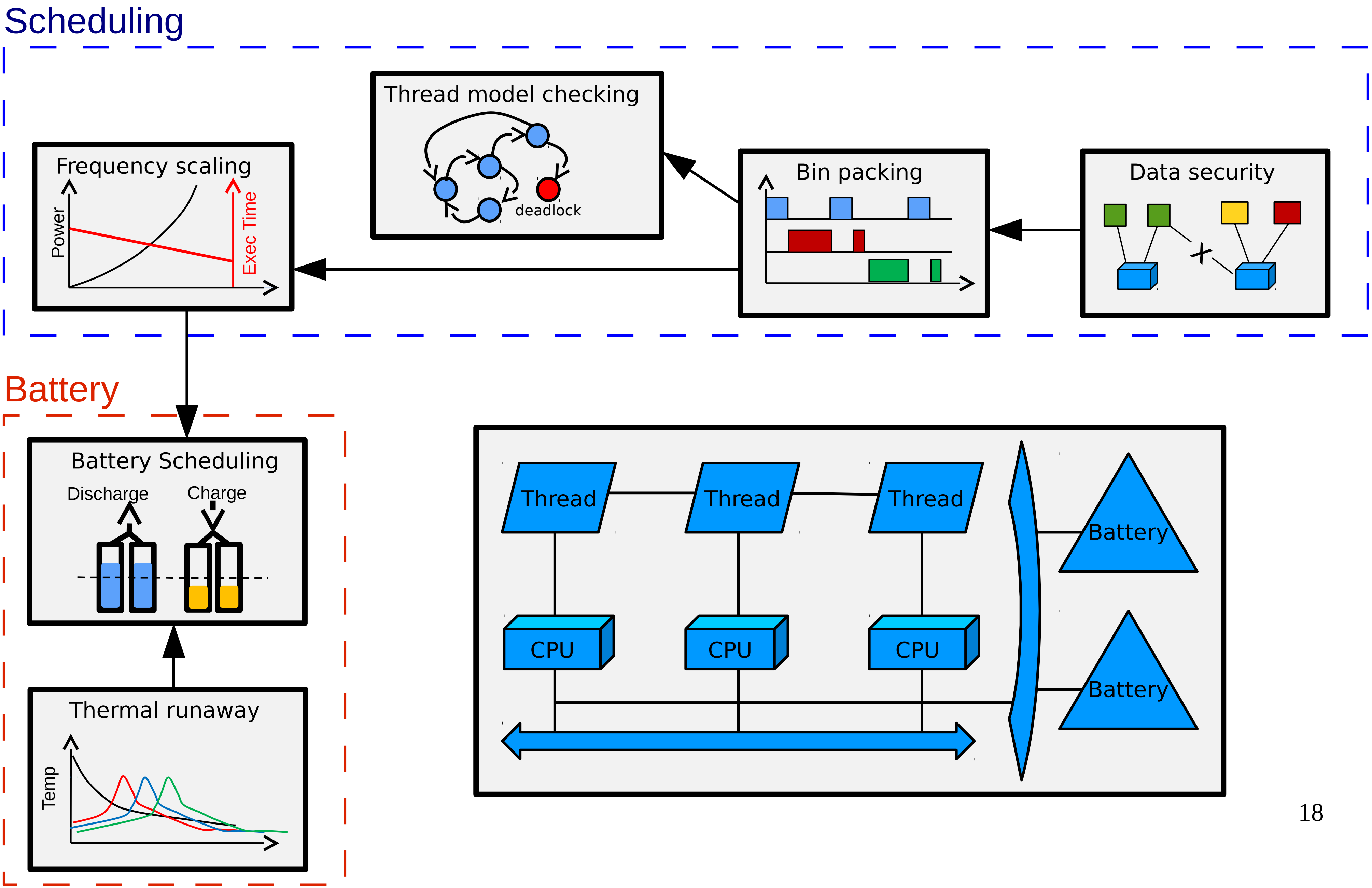
Analysis contract

- Given a domain $\boldsymbol{\sigma}$, *analysis contract* **C** is a tuple (**I**, **O**, **A**, **G**)
  - Inputs $\mathbf{I} \subseteq \mathcal{A} \cup \mathcal{S}$
  - Outputs $\mathbf{O} \subseteq \mathcal{A} \cup \mathcal{S}$
  - Assumptions $\mathbf{A} \subseteq \mathcal{F}_{\boldsymbol{\sigma}}$
  - Guarantees $\mathbf{G} \subseteq \mathcal{F}_{\boldsymbol{\sigma}}$
- Where:
  - $\mathcal{F}_{\boldsymbol{\sigma}} ::= \{\forall|\exists\}\ v_1..v_n \bullet \boldsymbol{\varphi} \ \mid\ \{\forall|\exists\}\ v_1..v_n \bullet \boldsymbol{\varphi} : \boldsymbol{\psi}$
  - $\boldsymbol{\varphi}$ is a static logical formula over $\mathcal{A}$ and $\mathcal{S}$
  - $\boldsymbol{\psi}$ is an LTL formula over $\mathcal{A}$, $\mathcal{S}$, and $\mathcal{R}$
- $\mathcal{F}_{\boldsymbol{\sigma}}$ semantics is given in a standard way
  - **:** means ⇒ in case of ∀, and ∧ in case of ∃

Outline

- Analysis integration problem
- Analysis contracts approach
  - Specification
  - **Verification**
- Experimental results

Contract I/O dependencies



Scheduling

Thread model checking

Frequency scaling

Bin packing

Data security

Battery

Battery Scheduling

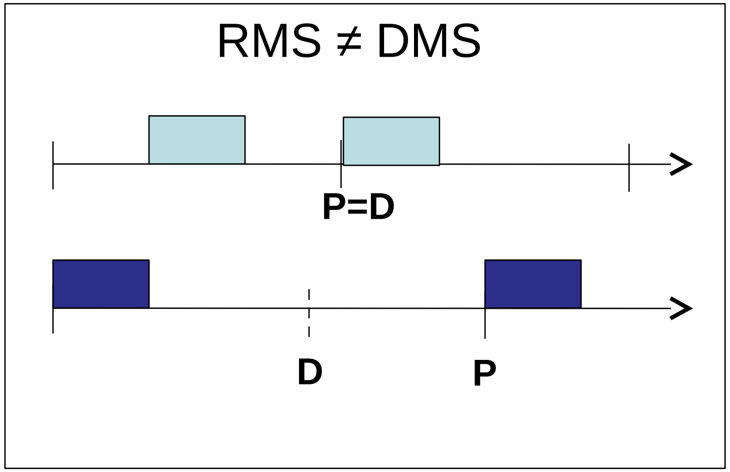
Thermal runaway

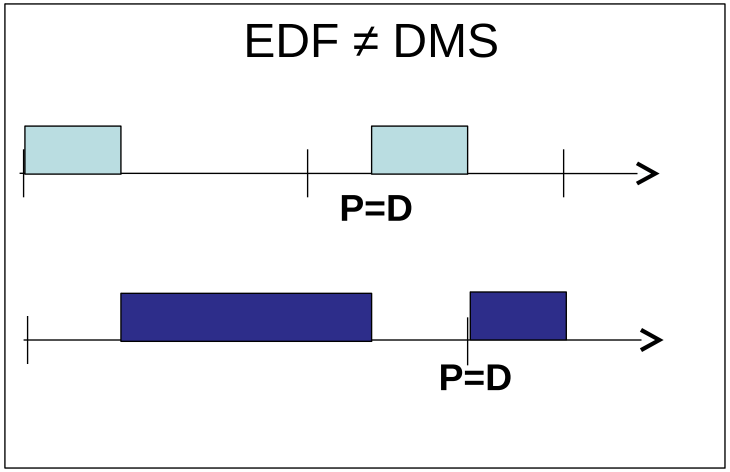
Frequency scaling assumption

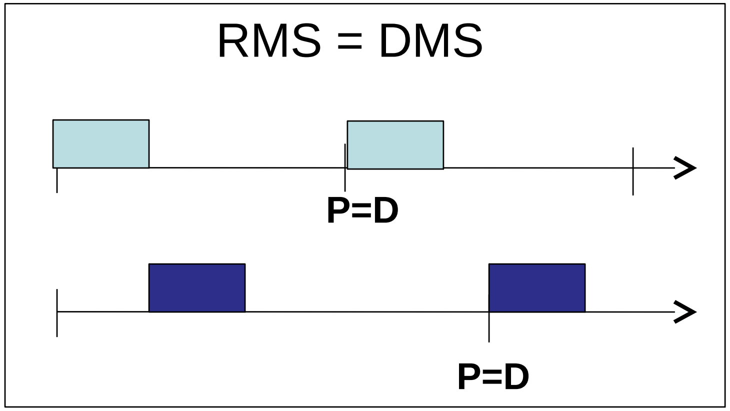
Behavioral equivalence to deadline-monotonic scheduling:

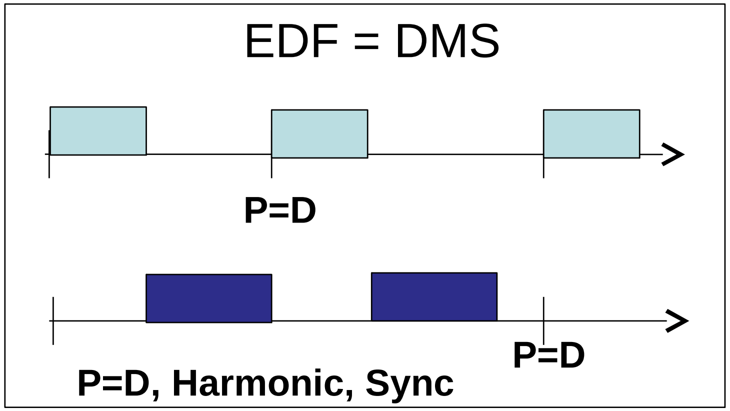
- $\forall\, t_1, t_2: \mathit{Threads} \bullet t_1 \neq t_2 \wedge \mathit{CPUBind}(t_1) = \mathit{CPUBind}(t_2)$:

  $G\ (\mathit{CanPrmpt}(t_1, t_2) \Rightarrow \mathit{Dline}(t_1) < \mathit{Dline}(t_2))$





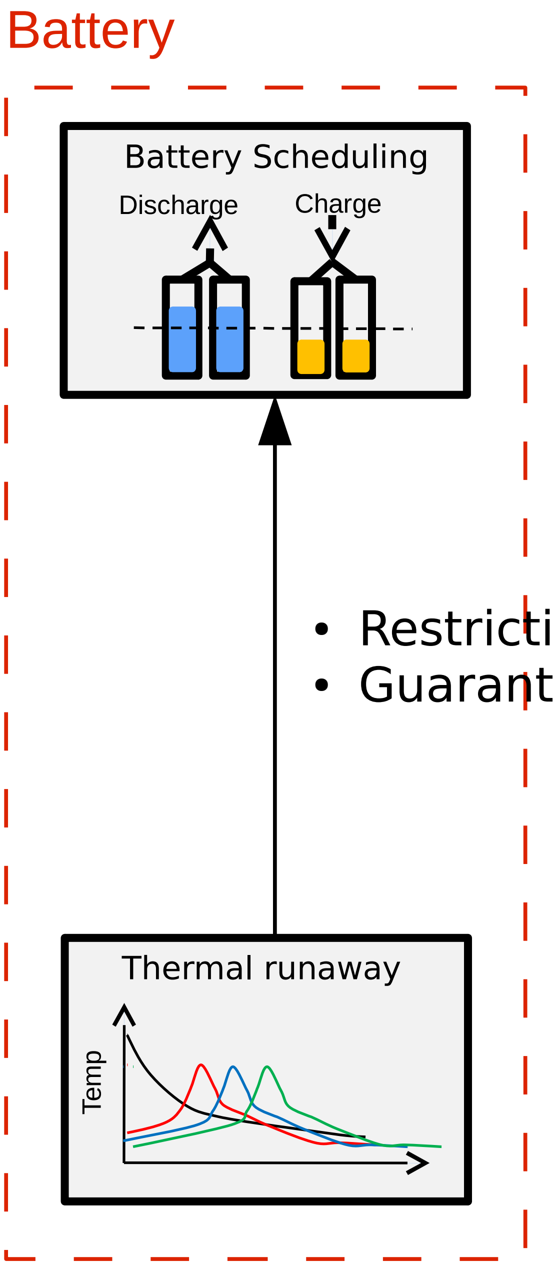




Assumption verification

- SMT solver finds solutions for static fragment $\boldsymbol{\varphi}$
  - $\forall$ t_1, t_2:*Threads* | $t_1 \neq t_2 \wedge$ *CPUBind*(t_1) = *CPUBind*(t_2)
- Model checking property $\boldsymbol{\psi}$ in a behavioral Promela model for each SMT solution:
  - G (*CanPrmpt*(t_1, t_2) $\Rightarrow$ *Dline*(t_1) < *Dline*(t_2))

Battery modeling



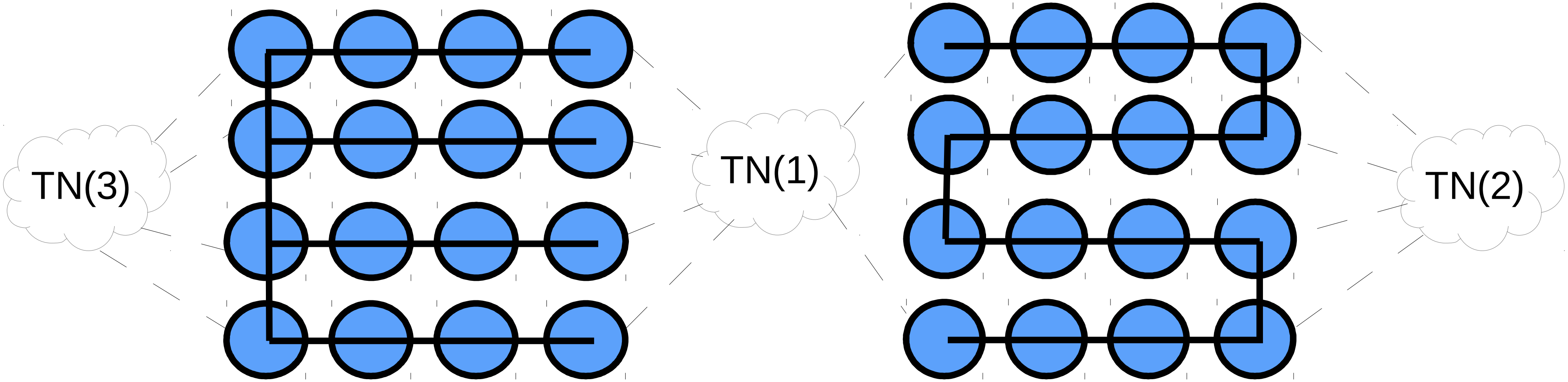
- Abstraction: circuits
- Selects a scheduler for cell connections
- Oblivious of heat: treats any configuration as acceptable heat-wise

- Restrictions on acceptable thermal configurations
- Guarantee: unacceptable ones don't occur

- Abstraction: geometry
- Simulates heat propagation
- Cannot scale to dynamic scheduling: simulates only fixed cell configurations

Battery scheduling guarantee

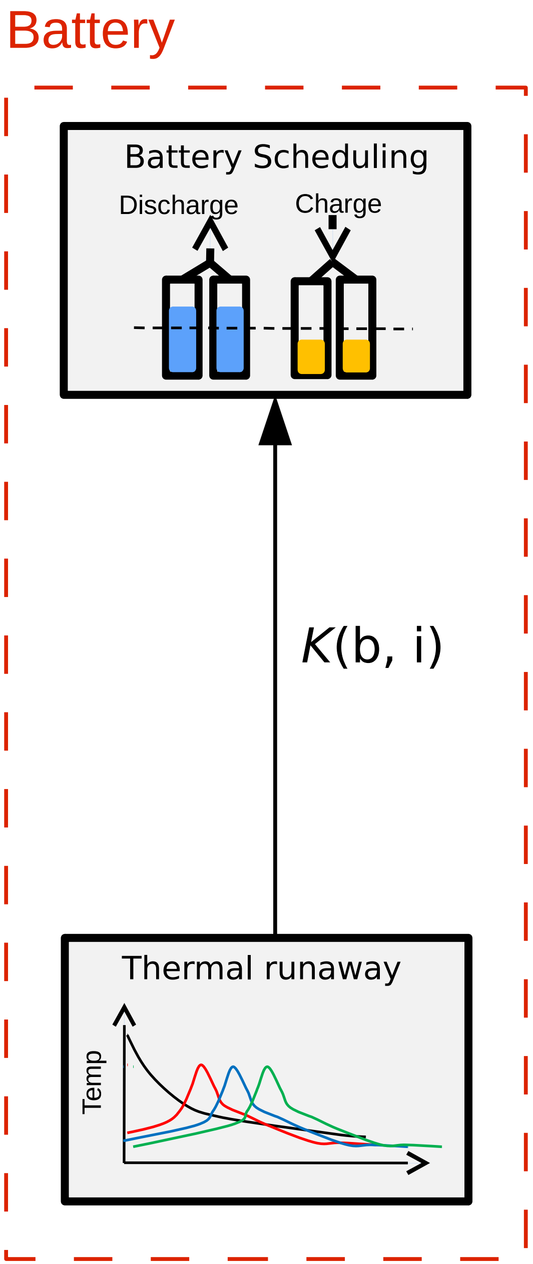
- “Bad” thermal configurations not reachable



- *TN*(b, i) ∈ $\mathcal{R}$ – number of cells in b with i thermal neighbors
- *K*(b, i) ∈ $\mathcal{S}$ – experimental coefficient for *TN*(b, i)
- Guarantee:

  ∀ b: *Batteries* • G ($\sum_{i=0..3}$ *K*(b, i)**TN*(b, i)) ≥ 0

Battery modeling



Selects a battery scheduler

Guarantee: $\forall$ b: *Batteries* • G ($\sum_{i=0..3}$ *K*(b, i)**TN*(b, i)) ≥ 0

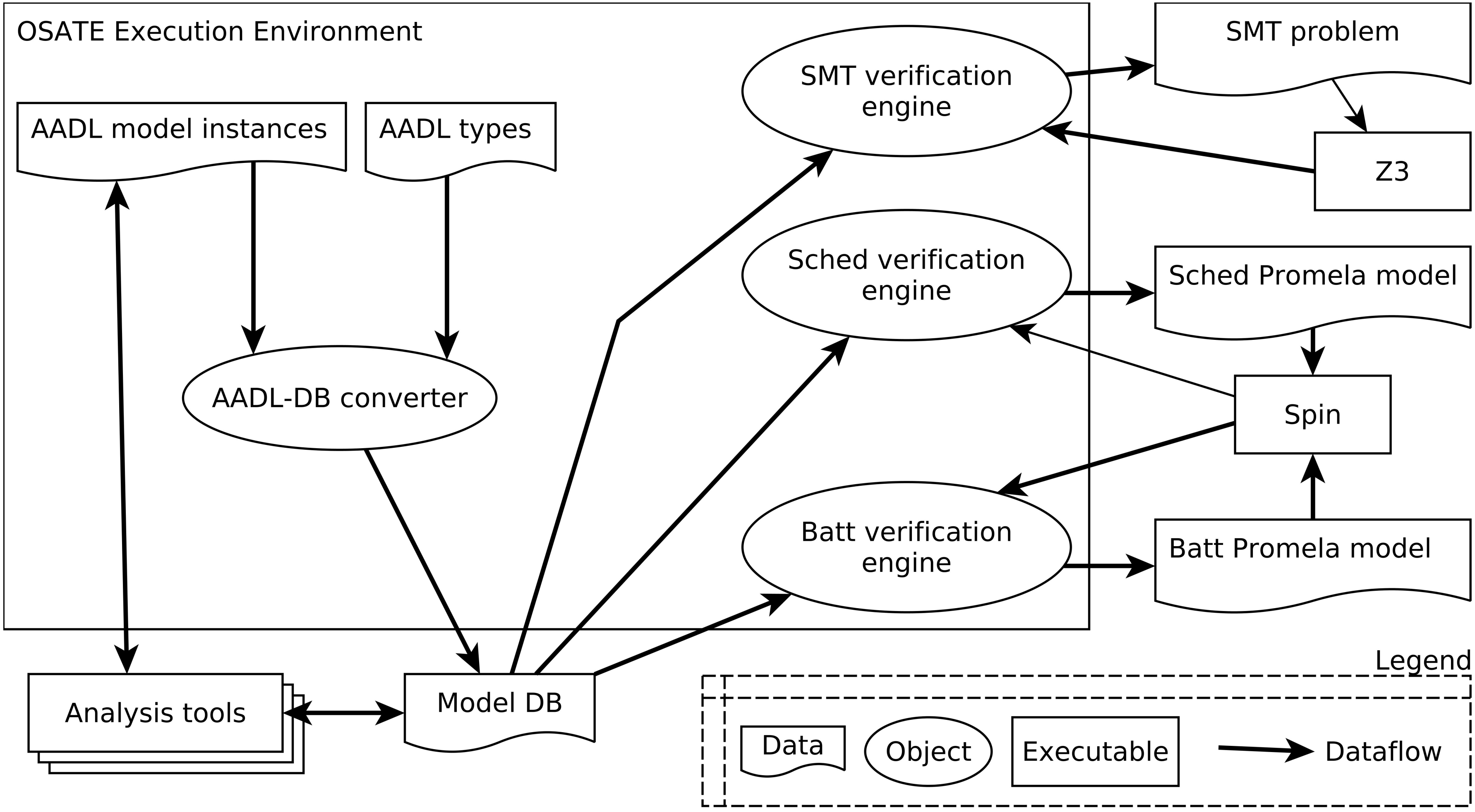
Verified with battery Promela/Spin model

Determines *K*(b, i) via simulation

Outline

- Analysis integration problem
- Analysis contracts approach
  - Specification
  - Verification
- **Experimental results**

Framework implementation



Scalability evaluation

- SMT solving typically takes less than 0.1 second
- Spin model checking times:

σ_{sched} :

Threads	(R/D)MS time	EDF time
3	0.01	0.01
4	0.01	0.52
5	0.07	33.4
6	0.37	2290.0
7	2.18	Out Mem
8	12.4	Out Mem
9	71.2	Out Mem
10	421	Out Mem
11	Out Mem	Out Mem

σ_{batt} :

Cells	FGURR time	FGWRR time	GPWRR time
9	0.13	0.15	0.15
12	0.61	2.34	3.94
16	44	31.4	127
20	1060	619	Out Mem
25	Out Mem	Out Mem	Out Mem

All times are in seconds

Summary

- Analysis integration is error-prone
  - Incorrect ordering
  - Violation of implicit assumptions
- Our solution:
  - Contract specification language
  - Contract verification algorithm
- Effective, extensible, and scalable
- Future work:
  - Assumptions behind $\mathcal{I}$ implementation
  - Analysis contracts for multiple views

Contracts

Security Analysis

- $An_{sec}\,.\,C\colon I = \{T, \mathit{ThSecCl}\}, O = \{\mathit{NotColoc}\}, A = \emptyset, G = \{g\}$
  - $g\colon \forall\, t_1, t_2 \cdot \mathit{ThSecCl}(t_1) \neq \mathit{ThSecCl}(t_2) \Rightarrow t_1 \in \mathit{NotColoc}(t_2)$

Multiprocessor scheduling: (Binpacking + scheduling)

- $An_{sched}.\,C\colon I = \{T, C, \mathit{NotColoc}, \mathit{Per}, \mathit{WCET}, \mathit{Dline}\}, O = \{\mathit{CPUBind}\}, A = \emptyset, G = \{g\}$
  - $g\colon \forall\, t_1, t_2 \cdot t_1 \in \mathit{NotColoc}(t_2) \Rightarrow \mathit{CPUBind}(t_1) \neq \mathit{CPUBind}(t_2)$

Frequency Scaling

- $An_{freqsc}.\,C\colon I = \{T, C, \mathit{CPUBind}, \mathit{Dline}\}, O = \{\mathit{CPUFreq}\}, G = \emptyset, A = \{a\}$
  - $a\colon \forall t_1, t_2 \cdot \mathit{CPUBind}(t_1) = \mathit{CPUBind}(t_2)\colon G(\mathit{CanPrmpt}(t_1, t_2) \Rightarrow \mathit{Dline}(t_1) < \mathit{Dline}(t_2)$

Model checking periodic program (REK):

- $An_{rek}.\,C\colon I = \{T, C, \mathit{Per}, \mathit{Dline}, \mathit{WCET}, \mathit{CPUBind}\}, O = \{\mathit{ThSafe}\}, G = \emptyset, A = \{a_1, a_2\}$
- $a_1\colon \forall t \cdot \mathit{Per}(t) = \mathit{Dline}(t),\;\; a_2\colon \forall t_1, t_2 \cdot G(\mathit{Canprmpt}(t_1, t_2) \Rightarrow G\ \neg \mathit{CanPrmpt}(t_2, t_1))$

Thermal runaway:

- $An_{therm}.\,C\colon I = \{B, \mathit{BatRows}, \mathit{BatCols}, \mathit{Voltage}\}, O = \{K\}, A = \emptyset, G = \emptyset$

Battery Scheduling

- $An_{bsched}.\,C\colon I = \{B, \mathit{BatRows}, \mathit{BatCols}\}, O = \{\mathit{BatConnSchedPol}, \mathit{HasReqLifetime}, \mathit{SeriqlReq}, \mathit{ParalRea}\}, A = \emptyset, G = \{g\}$
- $g\colon G(K(0) \times TN(0) + K(1) \times TN(1) + K(2) \times TN(2) + K(3) \times TN(3) \geq 0)$